LAUREA AMMATTIKORKEAKOULU Uuden edellä

Rappari 1.0 piirustuspöydältä sovellukseksi

Raportointisovelluksen kehitys Java-ohjelmointikielellä

Seppälä, Tiia

2013 Kerava

Laurea-ammattikorkeakoulu
Kerava

Rappari 1.0 piirustuspöydältä sovellukseksi

Raportointisovelluksen kehitys Java-ohjelmointikielellä

Tiia Seppälä
Tietojenkäsittelyn koulutusohjelma
Opinnäytetyö
Maaliskuu, 2013

Laurea-ammattikorkeakoulu
Kerava
Tietojenkäsittelyn koulutusohjelma

Tiivistelmä

Tiia Seppälä

Rappari 1.0 piirustuspöydältä sovellukseksi

Vuosi 2013 Sivumäärä 36

Tämän opinnäytteen tarkoituksena oli suunnitella ja luoda raportointityökalusovellus. Sovelluksen avulla käyttäjä voi ladata tietoja tietokannasta sekä luoda ja tallentaa taulukkomuotoisen raportin yhdessä sovelluksessa. Raportti tallennetaan Excel-tiedostoon.

Sovellus on toteutettu Java-ohjelmointikielellä. Toteutuksessa on myös käytetty JExcelAPI ja JDBC -kirjastoja. JExcelAPI-kirjaston avulla sovellus voi tallentaa tiedoston Excel-tiedostoon. JDBC-kirjaston avulla sovellus voi yhditäytyä tietokantaan ja hakea tietoja sieltä.

Sovelluksen kehityksessä on pyritty ottamaan käytettävyys mahdollisimman hyvin huomioon. Tämän toteuttamiseksi sovelluksen suunnittelun ja kehityksen apuna on käytetty metodeja, joita yleensä käytetään valmiin sovelluksen käytettävyyden arviointiin.

Lopputuloksena saatu sovellus täyttää kaikki vaatimusmäärittelyn asettamat vaatimukset. Kuitenkin jotkut sovelluksen elementit vaativat jatkokehitystä, jotta sovelluksen käytettävyys paranisi.

Asiasanat: Java, Excel, JExcelAPI, JDBC, sovelluskehitys, käytettävyys

Laurea University of Applied Sciences
Kerava
Business Information Technology

Abstract

Tiia Seppälä

Rappari 1.0 from drawing table into application

Year 2013 Pages 36

The purpose of this thesis was to design and create a reportage tool. With the application the user can download data from database, create and save a report within one application. The report is saved in Excel file.

The application was created with Java programming language. JExcelAPI and JDBC libraries were also used in the production. With JExcelAPI library the application can create a proper xls format file. With JDBC library the application can connect to a database and fetch data from there.

Usability was the most important element that the application had to have. Thus during the development some evaluation methods were used that are usually used to evaluate the usability of a ready application.

The finished application fulfilled all the requirements that the requirement specification stated. Although some elements need more development in order to improve usability.

Keywords: Java, Excel, JExcelAPI, JDBC, application development, usability

Sisällys

1 Johdanto 7
2 Projektin esittely 7
2.1 Toimeksianto 7
2.2 Vanha prosessi 8
3 Vaatimusmäärittely 9
3.1 Asiakasvaatimukset 9
3.1.1 Käyttöliittymän vaatimukset 9
3.1.2 Toiminnalliset vaatimukset 10
3.2 Käyttötarinat 11
3.2.1 Käyttötarina nro. 1 11
3.2.2 Käyttötarina nro. 2 12
4 Toiminnallinen määrittely 12
4.1 Tietojen lataus 14
4.2 Raportin luonti 14
4.2.1 Listan lisäys 14
4.2.2 Kaavan lisäys 14
4.2.3 Muut toiminnot 15
4.3 Exceliin vienti 16
5 Tekninen määrittely 16
5.1 Arkkitehtuurikuvaus 16
5.2 Swing-kirjasto 17
5.3 Käytetyt rajapinnat 17
6 Käyttöliittymän suunnittelu 18
6.1 Käytetyt tekniikat 18
6.2 Suunnittelu 19
6.2.1 Päänäkymä 19
6.2.2 Tietojen valinta -dialogi 21
6.2.3 Kaavan lisäys -dialogi 25
6.2.4 Navigointi 26
6.2.5 Virheilmoitukset 27
7 Toteutus 28
7.1 Käyttöliittymän ulkoasu 28
7.2 Taulukko 28
7.2.1 Solun ulkoasun muuttaminen 29
7.2.2 Virhetilanteet 30
7.3 Raportin tallennus 30
7.4 Kielitiedostot 31

8 Testaus 31
9 Yhteenveto 32
Lähteet 33
Kuvat 34
Liitteet 35

1 Johdanto

Tämän opinnäytetyön tavoitteena oli kehittää raportointityökalusovellus, jonka avulla tietojen lataus tietokannasta ja raportin luonti niiden pohjalta voitaisiin tehdä helposti samassa sovelluksessa. Sovellus on suunniteltu toimimaan tietyssä tietokannassa, mutta se olisi pienin muutoksin mahdollista saada toimimaan muissakin tietokannoissa.

Sovellus tehtiin toimeksiantona verkkopohjaisiin ratkaisuihin keskittyneelle yritykselle huomioiden erään yrityksen asiakkaan vaatimuksia. Yritys suunnittelee, kehittää ja ylläpitää verkkosivuja ja verkossa toimivia sovelluksia.

Sovellusta on kehitetty käytettävyyden näkökulmasta, jotta loppukäyttäjän käyttökokemus olisi mahdollisimman miellyttävä. Tähän on pyritty käyttämällä jo kehitysvaiheessa käytettävyyden arvioinnin tekniikoita, joita yleensä käytetään vasta valmiin sovelluksen arviointiin.

Tässä opinnäytteessä esitellään sovelluksen toteutuksen vaiheita vanhasta prosessista uuden sovelluksen toteutukseen. Aluksi esitellään, mistä lähtökohdista sovellusta lähdettiin kehittämään eli toimeksiannosta, sekä miten vanha prosessi toteutettiin. Tämän jälkeen esitellään eri määrittelyt, jotka ovat olleet sovelluksen suunnittelun ohjenuorana. Nämä määrittelyt ovat vaatimusmäärittely, toiminnallinen määrittely ja tekninen määrittely. Seuraavaksi esitellään käyttöliittymän suunnittelu ja sen apuna käytetyt tekniikat. Tämän jälkeen esitellään joitakin sovelluksen toteutuksen ratkaisuista sekä testauksessa käytetyt metodit.

2 Projektin esittely

Tässä luvussa esitellään, millaisista lähtökohdista sovellusta lähdettiin suunnittelemaan. Aluksi kerrotaan, miksi ja mitä varten sovellus kehitettiin. Tämän jälkeen kerrotaan tarkemmin vanhasta prosessista ja siihen liittyvistä ongelmista.

2.1 Toimeksianto

Yrityksen eräs asiakas jakaa fyysisiä mainoslehtiä Uudenmaan alueella. Yritys ylläpitää asiakkaan verkkosivuja. Näillä verkkosivuilla on mahdollista osallistua asiakkaan järjestämään kilpailuun, joka vaihtuu muutaman kerran kuukaudessa. Kilpailuun osallistutaan lomakkeen kautta, jossa kilpailuun osallistujan on annettava vastaus annettuun kysymykseen sekä joitain henkilökohtaisia tietoja, kuten nimi, paikkakunta, sukupuoli ja ikäryhmä. Lomakkeessa kysytään myös, mikä oli kilpailuun osallistuneen mielestä lehden kiinnostavin mainostaja.

Asiakas haluaa tietää, millaiset ihmiset ovat kiinnostuneet tietyistä mainostajista. Tätä varten yritys tuottaa asiakkaalle raportin, josta asiakkaan haluamat tiedot käyvät ilmi.

Työharjoittelussa ollessani sain tehtäväkseni tuottaa kyseisen raportin. Raporttiin haluttiin selvitys kilpailuun osallistuneiden paikkakunta- ja ikäjakaumasta. Aluksi tehtävä vaikutti helpolta, mutta pian huomasin, että prosessissa on parantamisen varaa. Prosessia hankaloittivat mm. tietojen lataaminen verkkosivuilta useassa erässä, listojen siirtäminen Excelistä toiseen leikkaa-liimaa taktiikalla sekä suurten kaavamäärien lisäys raahaamalla hiirellä.

Tästä sain idean sovellukseen, jolla olisi mahdollista ladata kaikki tarvittavat tiedot suoraan käsiteltäväksi helppokäyttöiseen sovellukseen ja jossa listat olisi suoraan käytettävissä ilman leikkaa-liimaa operaatiota ja suurten kaavamäärien lisäys kävisi helpommin. Raportti myös tallennettaisiin vanhaan tapaan Excel-muodossa. Sovellus poistaisi turhia välivaiheita ja yksinkertaistaisi raportin luontia huomattavasti.

2.2 Vanha prosessi

Vanha prosessi alkaa tietojen lataamisella asiakkaan verkkosivuilta. Verkkosivuilla on tätä varten oma toiminto. Käyttäjä valitsee, mitkä sarakkeet hän haluaa ladata. Tämän jälkeen hän kertoo, miltä aikaväliltä tiedot halutaan. Käyttäjän on mahdollista valita vaihtoehdoista: ”lataa kaikki”, ”viimeisimmät X riviä” ja ”rivit id väliltä X ja Y”. ”Lataa kaikki” –vaihtoehto ei ole käytännössä mahdollinen, sillä tietokannassa on syyskuusta lähtien ollut yli 16000 riviä dataa, ja yli 4000 rivin lataaminen aiheuttaa virhetilanteen verkkosivuilla, jolloin tiedot eivät lataudu. Käyttäjän ainoa vaihtoehto on täten tietojen lataaminen useassa erässä rivien id:den perusteella.

Käyttäjän saatua kaikki tiedot ladattua hänen on nyt yhdistettävä useassa tiedostossa sijaitsevat tiedot yhteen tiedostoon. Käytännössä käyttäjä kopioi ja liittää tietoja Excelistä toiseen. Kun käyttäjä on saanut kerättyä tiedot yhteen Excel-taulukkoon, raportin luonti voidaan aloittaa. Aluksi käyttäjä hakee toisesta Excel-taulukosta listan Suomen paikkakunnista ja postinumeroista. Käyttäjä kopioi kyseisen listan samaan taulukkoon, jossa data sijaitsee. Tämän jälkeen käyttäjä kirjoittaa listan ensimmäisen solun viereen ja kopioi koko listan pituudelle Excelin COUNTIF-kaavan, joka laskee, kuinka monta kertaa tietty postinumero esiintyy tiedoissa. Käyttäjä voi kopioida kaavan raahaamalla tai makron avulla. Tämän jälkeen käyttäjä hakee listan Suomen paikkakunnista ja kopioi sen taulukkoon. Tämän jälkeen käyttäjä lisää listan ensimmäisen solun viereen ja kopioi koko listan pituudelle SUMIF-kaavan, joka laskee yhteen niiden COUNTIF-kaavojen tulokset, joiden postinumerot liittyvät tiettyyn paikkakuntaan. Tämän jälkeen käyttäjä toistaa samankaltaisen prosessin ikäryhmien kohdalla, tosin käyttäjä tarvitsee vain ikäryhmät-listan ja COUNTIF-kaavat.

3 Vaatimusmäärittely

Vaatimusmäärittelyllä tarkoitetaan kokoelmaa vaatimuksista, jotka sovelluksen halutaan täyttävän. Sovelluksen katsotaan olevan valmis, kun se täyttää vaatimusmäärittelyn asettamat vaatimukset. Tässä luvussa esitellään asiakasvaatimukset, jotka asiakas haluaa sovelluksen täyttävän. Luvussa esitellään myös kaksi käyttötarinaa, joiden avulla havainnollistetaan, miten sovelluksen tulisi toimia. (Haikkala & Mikkonen 2011, 62-66.)

3.1 Asiakasvaatimukset

Asiakasvaatimuksilla tarkoitetaan niitä vaatimuksia, jotka tulevat suoraan asiakkaalta eli suoraan asiakkaiden tarpeista. Nämä asiakasvaatimukset on kirjoitettu jälkikäteen puhtaaksi, sillä sovelluksen kehitystä aloitettaessa vaatimukset olivat enemmänkin miellekartan muodossa. (Haikkala & Mikkonen 2011, 62.)

Projektissa asiakasvaatimuksia on kahdenlaisia: käyttöliittymän vaatimukset ja toiminnalliset vaatimukset. Käyttöliittymän vaatimukset ovat sovelluksen käyttöliittymän ominaisuuksiin kohdistuvia vaatimuksia. Toiminnalliset vaatimukset ovat suoraan sovelluksen toimintoihin liittyviä vaatimuksia.

3.1.1 Käyttöliittymän vaatimukset

Sovelluksella on oltava graafinen käyttöliittymä, jonka kautta käyttäjä voi käyttää sovellusta raportin luomiseen. Käyttöliittymän kautta käyttäjän tulisi siis pystyä lataamaan tiedot, luomaan raportin ja tallentamaan raportin Excel-tiedostoon.

Käyttöliittymän on oltava helppokäyttöinen, jotta sovelluksen käyttö olisi mahdollisimman nopeaa ja miellyttävää. Helppokäyttöisyyteen vaikuttavat monet asiat, mutta tässä tapauksessa tähän tavoitteeseen päästään, kun käyttöliittymä täyttää seuraavat vaatimukset: toiminnot ovat helposti saatavilla, toiminnot toimivat loogisesti ja käyttöliittymä on selkeä. (Kuoppala, Parkkinen, Sinkkonen & Vastamäki 2006, 194.)

Raportin luontia varten sovelluksen päänäkymässä on oltava taulukko, johon tiedot ladataan tietokannasta ja raportti luodaan. Näiden ladattujen tietojen olisi hyvä erottua muista raportin tiedoista, esimerkiksi solujen taustavärin perusteella. Tämä erottelu ei kuitenkaan saa periytyä xls-tiedostoon.

Sovelluksen on tunnistettava seuraavat Excelin kaavat: SUM, SUMIF, COUNT ja COUNTIF. Lisäksi sovelluksessa pitäisi olla kaava, jonka avulla käyttäjä saisi helposti laskettua esimerkiksi ikäjakaumien prosenttiosuuden. Tämän kaavan nimi on PERCENT, sen täytyisi laskea prosentti kaavalla jaettava/jakaja*100.

Sovelluksen on myös ratkaistava nämä kaavat ja näytettävä käyttäjälle kaavojen lopputulokset, jotta käyttäjä näkee suurin piirtein onko kaavat oikein. Excelin kaavojen kanssa saattaa olla joskus ongelmatilanteita, esimerkiksi ikuinen silmukka, jossa kaksi SUM-kaavaa laskee toisiaan koko ajan yhteen. Tästä syystä sovelluksen on tunnistettava yleisimmät kaavojen virheet.

3.1.2 Toiminnalliset vaatimukset

Toiminnallisilla vaatimuksilla tarkoitetaan toimintoja, jotka sovellukseen tulee osata. Tässä kappaleessa kerrotaan toiminnoista vain käyttäjän vaatimusten näkökulmasta. Seuraavassa luvussa kerrotaan syvällisemmin sovelluksen toimintojen teknisestä toteutuksesta.

Sovelluksessa on oltava toiminto jonka avulla käyttäjä voi ladata haluamiansa tietoja tietokannasta käyttöönsä. Tämän toiminnon tulisi toimia erillisen dialogin kautta. Dialogissa käyttäjä valitsisi, mitkä tiedot hän haluaa ladata ja käynnistäisi tietojen latauksen.

Käyttäjän on pystyttävä dialogissa valitsemaan, mitkä sarakkeet ja miltä aikaväliltä sarakkeet ladataan. Sarakkeilla tarkoitetaan erilaisia kilpailuun osallistuneiden tietoja, kuten nimi tai paikkakunta. Aikavälillä tarkoitetaan sitä, millä aikavälillä kilpailuun on osallistuttu.

Käyttäjän on myös pystyttävä asettamaan ehtoja sille, millaisia tietoja ladataan. Käyttäjän tulisi esimerkiksi pystyä valitsemaan, että vain niiden kilpailuun osallistuneiden tiedot ladataan, joiden ikä osuu 35-55 ikähaarukkaan. Ehtojen valinnan vaihtoehdon ei tarvitse olla välttämättä esillä, joten ehtojen valinta voidaan tehdä myös erillisessä dialogissa.

Sovelluksessa on oltava raportin luontia varten erilaisia toimintoja, jotta raportin luonti olisi helppoa. Nämä toiminnot ovat listan lisäys, kaavan lisäys ja manuaalinen kirjoittaminen.

Listan lisäys -toiminnolla tarkoitetaan sellaista toimintoa, jonka avulla käyttäjä voi lisätä raporttiin listan helposti, esimerkiksi kaavaa varten. Listat ovat luetteloita käyttäjän kannalta oleellisista tiedoista raportin luontia varten. Lista Suomen paikkakunnista on esimerkiksi yksi tällainen lista. Käyttäjällä on myös oltava mahdollisuus muokata näitä listoja.

Kaavan lisäys -toiminnolla tarkoitetaan toimintoa, jonka avulla käyttäjä voi lisätä kerralla suuren määrän kaavoja tietylle alueelle. Toiminto toimisi niin, että käyttäjä antaa sovellukselle tarvittavat tiedot ja sovellus tämän jälkeen lisää kaavat haluttuihin soluihin. Toiminnon on muutettava kaava automaattisesti sopimaan eri soluihin.

Manuaalinen kirjoitus -toiminnolla tarkoitetaan sitä, että käyttäjän on pystyttävä kirjoittamaan manuaalisesti taulukkoon. Käyttäjän on pystyttävä tekemään tämä milloin tahansa. Käyttäjä ei kuitenkaan saisi pystyä kirjoittamaan ladattujen tietojen päälle.

Raportin luonnin jälkeen käyttäjän on pystyttävä tallentamaan raportti xls-tiedostoon. Käyttäjällä on oltava mahdollisuus valita tiedoston nimi ja sijainti, samaan tapaan kuin muissakin sovelluksissa. Kaavojen on myös toimittava tallennetussa tiedostossa oikeaa Exceliä käytettäessä.

3.2 Käyttötarinat

Tässä kappaleessa esitetään kaksi käyttötarinaa, joita on käytetty sovelluksen suunnittelun ja toteutuksen apuna. Käyttötarinoita käytetään yleisesti erilaisten projektien suunnittelussa ja toteutuksessa. Käyttötarinoiden avulla voidaan helposti avata projektin tavoitteet, eli tässä tapauksessa niiden avulla kerrotaan miten vaatimusten tulisi täyttyä käytännössä. (Kuoppala ym. 2006, 25.)

3.2.1 Käyttötarina nro. 1

Heikki haluaa selvittää kilpailuun osallistuneiden ikä- ja sukupuolijakauman ja sen, keitä mainostajia kilpailuun osallistuneet ovat pitäneet kiinnostavimpina lehdessä 14/2012.

Heikki avaa sovelluksen ja aktivoi toiminnon ”lataa kilpailuun osallistuneiden tietoja”. Heikki valitsee ladattavissa olevista osallistuneiden tiedoista tiedot ikäryhmä, sukupuoli sekä kiinnostavin mainostaja. Tämän jälkeen Heikki syöttää aikavälin, jolta hän haluaa edellä mainitut tiedot ladata. Alkupäivämäärä on lehden ilmestymispäivä ja päättymispäivämäärä on päivä ennen lehden 15/2012 ilmestymistä. Heikki aloittaa tietojen latauksen, kun nämä tiedot on syötetty.

Latauksen valmistuttua Heikki näkee ladatut tiedot taulukossa sovelluksen päänäkymässä. Ladattuja tietoja on kolme saraketta ja n. 500 riviä. Aluksi Heikki haluaa selvittää ikäjakauman. Heikki valitsee, mistä taulukon solusta hän haluaa ikäryhmät-listan alkavan ja tämän jälkeen hän antaa komennon, joka lisää listan taulukkoon. Listan lisäyksen jälkeen Heikki haluaa laskea, kuinka moni eri ikäryhmistä oli osallistunut kilpailuun, joten Heikki aktivoi kaa-

van lisäys -toiminnon. Heikki haluaa käyttää toimintoa lisätäkseen useamman COUNTIF-kaavan samalla kertaa. Heikki antaa kaavan vaatimat tiedot sovellukselle, ja sovellus lisää kaavat taulukkoon. Heikki toistaa saman loppujen tietojen kohdalla.

Heikki on nyt saanut raporttinsa valmiiksi, joten on aika tallentaa raportti Excel-tiedostoon. Heikki aktivoi toiminnon, jonka avulla tallennus toteutetaan. Heikki valitsee tiedoston sijainnin ja nimen ja tämän jälkeen hän painaa ”tallenna”-nappia. Sovellus antaa varoituksen, koska Heikillä on valitussa sijainnissa jo entuudestaan samanniminen tiedosto, ja kysyy haluaako Heikki korvata tiedoston. Heikki huomaa syöttäneensä väärän nimen ja vastaa sovellukselle ”ei” ja korjaa nimen oikeaksi. Tämän jälkeen Heikki yrittää uudestaan tallentaa tiedostoa ja tällä kertaa tallennus onnistuu hyvin. Heikki on saanut raporttinsa tehtyä, joten Heikki sulkee sovelluksen.

3.2.2 Käyttötarina nro. 2

Heikki haluaa selvittää millaiset mainostajat lehdestä 14/2012 kiinnostivat lehden jakelualueella sijaitsevia 18-35 -vuotiaita kilpailuun osallistuneita.

Heikki käynnistää sovelluksen ja toimii tietojen latauksen kanssa muuten samoin kuin ensimmäisessä käyttötarinassa, mutta tällä kertaa syötettyään tiedot halutuista tiedoista ja aikavälistä, Heikki klikkaa ”Valitse ehtoja” –painiketta, jolloin aukeaa uusi ikkuna. Tässä uudessa ikkunassa Heikki valitsee ehdot. Ehdot valitaan erilaisista vaihtoehdoista. Ensin Heikki valitsee paikkakuntaan liittyvät ehdot: Heikki avaa paikkakuntaosion ja valitsee ehdoksi jakelualue. Tämän jälkeen Heikki valitsee ikäryhmäosiosta ehdot 18-25 ja 25-35. Tämän jälkeen Heikki sulkee ehtojen valinta -ikkunan ja aloittaa tietojen latauksen.

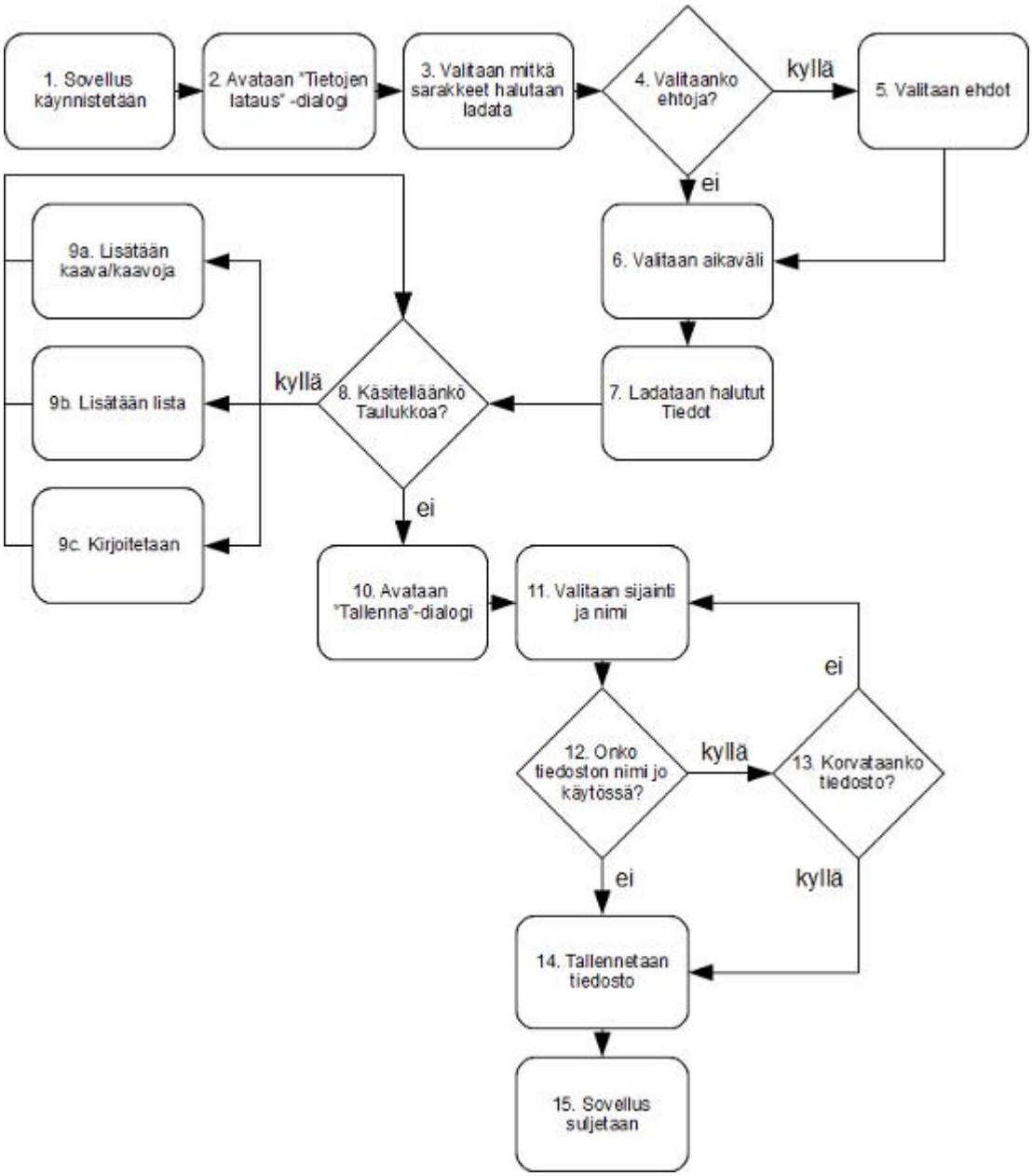
Tietojen latauduttua Heikki lisää raporttiin listan lehden 14/2012 mainostajista ja lisää COUNTIF-kaavat kuten edellisessä käyttötarinassa. Heikki haluaa myös selvittää mainostajien prosenttijakauman. Tämän Heikki saa helposti selville sovelluksen PERCENT-kaavan avulla. Aluksi Heikin on kuitenkin selvitettävä, kuinka monta riviä tietoja on, ja tämä saadaan helposti laskemalla SUM-kaavalla yhteen se, kuinka moni piti kutakin mainostajaa kiinnostavimpana. Tämän jälkeen Heikki voi lisätä PERCENT-kaavan jokaisen mainostajan jälkeen, käyttämällä SUM-kaavalla saatua tulosta jakajana ja COUNTIF-kaavoilla saatuja tuloksia jaettavina. Raportti on tämän jälkeen valmis ja Heikki tallentaa raportin xls-tiedostoon.

4 Toiminnallinen määrittely

Tässä luvussa esitellään toiminnot, joilla vastataan vaatimusmäärittelyn asettamiin toiminnallisuuteen liittyviin vaatimuksiin. Luvussa selvitetään myös tarkemmin se, miten toimintoja

käytetään ja miten ne on toteutettu. Toiminnot esitellään siinä järjestyksessä, jossa käyttäjä käyttäisi niitä raporttia luodessaan. Nämä toiminnot voidaan periaatteessa jakaa kolmeen kategoriaan; tietojen lataus, raportin luonti ja tiedostoon vienti. Tietojen lataus ja tiedostoon vienti -kategoriat ovat tosin samalla itse toiminnot.

Kuvassa 1 esitellään vaatimusmäärittelyn ja käyttötarinoiden pohjalta koottu toimintakaavio sovelluksen käytöstä (kuva 1). Siinä on pyritty havainnollistamaan sovelluksen käyttöä mahdollisimman yksinkertaisesti ja loogisesti.



Kuva 1: Toimintakaavio sovelluksen toiminnoista

4.1 Tietojen lataus

Tämä toiminto kattaa kuvan 1 kohdat 1-7 (kuva 1). Ennen kuin tietoja voidaan ladata, käyttäjän on kerrottava sovellukselle, millaisia tietoja hän haluaa ladata. Tämä tapahtuu erillisessä dialogissa (kuva 1, kohta 2). Dialogissa käyttäjä voi valita ladattavat sarakkeet eli kilpailuun osallistuneiden tiedot. Käyttäjä voi myös määrittää aikavälin, jolta tietoja haetaan. Halutessaan käyttäjä voi myös määritellä ehtoja ladattaville tiedoille (kuva 1, kohta 5). Käyttäjä voi esimerkiksi valita, että ladataan niiden osallistujien tiedot, jotka osuvat ikähaarukkaan 18-35. Kun käyttäjä on mielestään antanut kaikki tarvittavat tiedot, hän aloittaa tietojen latauksen (kuva 1, kohta 7). Kun tiedot on ladattu, ne viedään muokkausnäkymän taulukkoon.

Tietojen lataus on käytännössä sovelluksen oleellisin toiminto, pääosin kahdesta syystä. Ensimmäiseksi, mahdollisuus ladata kaikki rivit tai rajata rivit päivämäärän mukaan pelkästään tuovat lisäarvoa käyttäjälle. Toiseksi, mahdollisuus ladata rivejä tietyin ehdoin antaa käyttäjälle paremmat mahdollisuudet tutkia esimerkiksi sitä, kiinnostavatko mainokset haluttua kohderyhmää.

4.2 Raportin luonti

Tämä kategoria kattaa kuvan 1 kohdat 8-9c (kuva 1). Muokkausnäkymässä käyttäjällä on käytettävissään erilaisia toimintoja helpottamaan raportin luontia. Kaksi tärkeintä toimintoa ovat listan lisäys ja kaavan lisäys.

4.2.1 Listan lisäys

Listat ovat luetteloita raportin kannalta oleellisista tiedoista. Näitä listoja ovat mm. listat eri lehtien mainostajista ja ikäryhmistä. Listojen lisäyksen on tapahduttava helposti, jotta sovellus olisi helppokäyttöinen tältä osalta.

Toiminto toimii niin, että käyttäjä valitsee, mistä taulukon solusta lista alkaa ja tämän jälkeen hän valitsee, minkä listan hän haluaa lisätä. Iso osa listoista on yhden sarakkeen levyisiä, joten ne lähtevät aloitussolusta suoraan alaspäin. Osa listoista voi kuitenkin olla useamman sarakkeen levyisiä, joten nämä listat lähtevät alas ja oikealle aloitussolusta.

4.2.2 Kaavan lisäys

Käyttäjä voi lisätä raporttiin kaavoja joko manuaalisesti kirjoittamalla tai erillisen dialogin kautta. Dialogin kautta käyttäjä voi lisätä helposti suurenkin määrän kaavoja kerralla.

Dialogissa käyttäjä valitsee kaavan, jonka hän haluaa lisätä jollekin alueelle. Käyttäjä voi valita alueeksi koko sarakkeen tai rajatun alueen. Yksittäisiin soluihin tätä toimintoa ei sovelleta, vaan käyttäjän on kirjoitettava kaava silloin manuaalisesti. Lisäksi koko sarakkeelle lisäys -ominaisuus on rajattu vain niin pitkälle, kuin kaavan vaatimia ehtoja riittää. Toisin sanoen, jos kaavan ehtosolussa ei ole sisältöä, kaavaa ei lisätä.

Käyttäjän on myös annettava kaavan tarvitsemat tiedot; esimerkiksi kaava COUNTIF vaatii tiedon alueesta, jolta lasketaan ehdon täyttäviä soluja sekä tiedon siitä, missä solussa ehto sijaitsee. Toisin kuin Excelissä, tässä sovelluksessa ehdoksi käy vain solu, sillä sovellus on tarkoitettu sellaiseen käyttöön.

Sovellus muuttaa kaavat automaattisesti sopimaan jokaiseen soluun, johon niitä on lisätty dialogia käyttämällä. Kuvassa 2 on esimerkkitapaus, jossa käyttäjä on lisännyt dialogin kautta useamman COUNTIF-kaavan allekkain sarakkeeseen kolme (kuva 2). COUNTIF-kaava kirjoitetaan muotoon countif(alue;ehto). Alueella tarkoitetaan niitä soluja, joiden arvoja verrataan ehtoon. Ehdolla tarkoitetaan sitä solua, jonka arvoon alueen solujen arvoja verrataan.

Käyttäjä on siis antanut sovellukselle tiedot, että hän haluaa lisätä COUNTIF -kaavan sarakkeeseen kolme. Tämän lisäksi hän on antanut alueeksi koko sarakkeen nolla ja ehdoksi sarakkeen kaksi. Kaavat on lisätty koko sarakkeen kolme laajuisesti, niin kauan kun sarakkeessa kaksi on ollut sisältöä vastaavalla rivillä. Toisin sanoen, kun toiminto aloittaa kaavojen lisäämisen soluun 3,0 sovellus laittaa ehdon soluksi 2,0. Kun sovellus siirtyy soluun 3,1 ehdon solu on 2,1 jne. Alue ei kuitenkaan muutu. Kaavojen lisäys on kuitenkin rajoitettu vain päällekkäiseen lisäykseen, sillä kaikki listat on myös rajoitettu pystysuuntaan.

	0	1	2	3
0	25-35		18-25	=countif(0;2,0)
1	45-55		25-35	=countif(0;2,1)
2	18-25		35-45	=countif(0;2,2)
3	18-25		45-55	=countif(0;2,3)

Kuva 2: Esimerkki useamman kaavan lisäyksen lopputuloksesta

4.2.3 Muut toiminnot

Käyttäjä voi muokata raporttia myös kirjoittamalla manuaalisesti taulukkoon. Tämä on enemmänkin ominaisuus kuin toiminto. Sovellus tunnistaa manuaalisesti kirjoitetun kaavan automaattisesti ja käsittelee solun sisältöä siten.

Muokkausnäkymän taulukon koko on rajallinen, joten sovelluksessa on toiminnot, joiden avulla käyttäjä voi lisätä taulukkoon sarakkeita ja rivejä. Sarakkeet ja rivit lisätään taulukon viimeisten sarakkeiden ja rivien perään.

4.3 Exceliin vienti

Tämä toiminto käsittää kuvan 1 kohdat 10-15 (kuva 1). Kun käyttäjä tyytyväinen raporttiin, hän voi viedä sen xls-tiedostoon. Tallennus tapahtuu erillisen dialogin kautta. Tämän dialogin kautta käyttäjä valitsee tiedoston sijainnin ja nimen.

Toisinaan on sellaisia tilanteita, jossa käyttäjän valitsema tiedoston nimi on jo käytössä. Tässä tilanteessa sovellus kysyy käyttäjältä, haluaako hän korvata vanhan tiedoston uudella. Käyttäjä voi valita raportille uuden sijainnin tai nimen, jos hän ei halua korvata vanhaa tiedostoa.

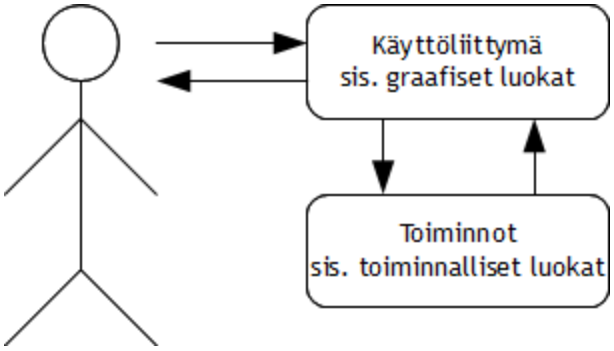
5 Tekninen määrittely

Tässä luvussa selostetaan yleisellä tasolla sovelluksen rakennetta ja sen toteutuksessa käytettyjä ratkaisuja. Aluksi esitellään sovelluksen arkkitehtuurikuvaus hyvin yksinkertaisella tasolla. Tämän jälkeen kerrotaan käyttöliittymän toteutukseen käytetystä Swing-kirjastosta. Lopuksi kerrotaan, mitä erillisiä kirjastoja sovellukseen on liitetty, sillä kaikkia käytettyjä kirjastoja Javassa ei ole valmiina.

5.1 Arkkitehtuurikuvaus

Sovelluksessa on kahden tyyppisiä luokkia: graafiset luokat ja toiminnalliset luokat. Graafiset luokat ovat sovelluksen luokkia, jotka sisältävät ainoastaan grafiikkaan vaikuttavia komponentteja. Näiden luokkien tehtävänä on huolehtia sisällön näyttämisestä käyttäjälle eli käyttöliittymän tuottaminen. Toiminnallisten luokkien tehtävänä on suorittaa sovelluksen eri toiminnot.

Sovelluksen käyttö tapahtuu graafisen käyttöliittymän kautta, jonka piirtäminen tapahtuu graafisten luokkien avulla. Näiden luokkien sisältä puolestaan kutsutaan toiminnallisten luokkien metodeja. Graafisissa luokissa on pyyntöjä, joilla laukaistaan toiminnallisten luokkien toimintoja. Toisin sanoen graafinen luokka lähettää toiminnalliselle luokalle pyynnön, että suorita toiminto X. Toiminnallinen luokka suorittaa toiminnon X ja lähettää graafiselle luokalle tiedon, että toiminto X on suoritettu. Tämän jälkeen graafinen luokka näyttää käyttäjälle toiminnon tuloksen. (Ks. kuva 3)



Kuva 3: Sovelluksen arkkitehtuuri

5.2 Swing-kirjasto

Kehitystyön alussa graafisen käyttöliittymän toteutukseen oli valittavissa Swing- ja AWT-kirjasto. Molempien kirjastojen komponentit on tarkoitettu sovelluksen graafisen käyttöliittymän luontiin. (Kosonen, Peltomäki & Silander 2007, 262-263.)

Käyttöliittymän toteutukseen valittiin Swing-kirjasto, koska se sisältää monia etuja AWT:hen nähden. Näistä tärkeimpinä mainittakoon mm. se, että AWT tuhlaa järjestelmän resursseja Swingiin nähden, eikä sillä ole omaa taulukko-komponenttia, jolloin se täytyisi tehdä itse. Resurssien tuhlaamisella tarkoitetaan sitä, että AWT syö tietokoneen muistia turhan paljon. Tämä on ongelmallista, sillä jos sovellus käyttää liikaa muistia, käyttäjän tietokoneesta voi loppua muisti, jolloin käyttäjän kone hidastuu. (Kosonen ym. 2007, 275; Oracle 2010a; Oracle 2010b.)

Taulukko-komponentin puuttuminen ei sinänsä ole ongelma toteutuksen näkökulmasta, sillä sen voi aina tehdä itse. Kuitenkin tällaisen taulukon toteutus olisi ollut sen verran aikaa vievää, että projektin aikataulu olisi pettänyt lähes varmasti. Kun huomioidaan myös edellä mainittu resurssien syönti, AWT:n käyttö sovelluksen toteutukseen olisi ollut kannattamatonta.

5.3 Käytetyt rajapinnat

Sovellus käyttää erillisiä kirjastoja, jotka sisältävät sovelluksen kannalta oleellisia rajapintoja eri toimintojen ohjelmointiin. Ilman näitä rajapintoja sovellus ei pystyisi käyttämään tietokantoja tai kirjoittamaan Excel-tiedostoon.

MySQL Connector/J on MySQL:n tarjoama virallinen JDBC tyypin 4 ajuri. Tässä sovelluksessa on käytetty MySQL Connector/J versiota 5.1.13. MySQL Connector/J tarjoaa JDBC ohjelmointirajapinnan, jonka avulla voidaan käyttää tietokantoja kokonaan Javalla. JDBC:n avulla sovellus ottaa yhteyden tietokantaan ja suorittaa SQL-kyselyt. (Kosonen ym. 2007, 531; Oracle 2012.)

Java Excel API on avoimen lähdekoodin ohjelmointirajapinta Javalle. Tämän rajapinnan avulla voidaan lukea ja kirjoittaa Microsoftin Excel-tiedostoja. Sovellus käyttää tätä kirjastoa raporttien tallennukseen. (JExcelApi 2012.)

Java Excel API rajapinta toimii siten, että sovellus kertoo rajapinnalle solun sijainnin, tekstin ja tyypin, esimerkiksi solun sijainti on A5, teksti on 38 ja tyyppi on numero. Tämän jälkeen rajapinta kirjoittaa sovelluksen antamien tietojen perusteella tiedot Excel-tiedostoon.

6 Käyttöliittymän suunnittelu

Tässä luvussa kerrotaan sovelluksen käyttöliittymän suunnittelusta. Kappaleessa 6.1 kerrotaan, mitä tekniikoita ja lähteitä käyttöliittymän suunnittelun apuna on käytetty. Kappaleessa 6.2 esitellään sovelluksen käyttöliittymää ja ratkaisujen perusteluita.

Käyttöliittymän suunnittelussa on pyritty ottamaan käytettävyys huomioon mahdollisimman hyvin. On tärkeää, että sovelluksen käyttöliittymä on toimiva ja hyvin suunniteltu, jotta käyttäjän olisi miellyttävä käyttää sitä. Huonosti suunniteltu ja toteutettu käyttöliittymä johtaa käyttäjän turhautumiseen ja edelleen negatiiviseen käyttökokemukseen. Käyttäjän negatiiviset tunteet vaikuttavat oleellisesti käyttäjän tehokkuuteen. (Kuoppala ym. 2006, 219.)

6.1 Käytetyt tekniikat

Tässä kappaleessa esitellään ne tekniikat, joita on käytetty käyttöliittymän suunnittelun apuna. Aluksi esitellään heuristista arviointia ja tämän jälkeen esitellään kirjalähteet.

Heuristista arviota käytetään käytettävyyden arvioinnin apuna. Yleensä tämä arviointi tehdään jonkun ulkopuolisen toimesta sovelluksen valmistumisen jälkeen, mutta tätä menetelmää voidaan käyttää myös käyttöliittymän suunnittelun tukena.

Heuristisessa arvioinnissa käytetään listoja säännöistä ja ohjeista, joita käyttöliittymän tulisi noudattaa. Nielsenin lista (Liite 1) on yksi käytetyimmistä käytettävyyden heuristisen arvioinnin apuvälineistä. Nielsenin listasta on monia eri versiota ja tämän sovelluksen kehittämisessä on käytetty kirjan ”Käytettävyys, suunnittelu ja arviointi” versiota, joka on vapaa käännös alkuperäisestä Nielsenin listasta. (Kuutti 2003, 49.)

Heuristinen arviointi toteutetaan niin, että arvioija käyttää sovellusta ja kirjaa havaitsemansa virheet muistiin ja sen mitä listan kohtaa virhe koskee. Tällä tavoin havaitaan monet käytettävyyden virheet, mutta kaikkia virheitä ei kuitenkaan voida havaita. (Kuutti 2003, 48.)

Käyttöliittymän suunnittelun apuna on myös käytetty kahta kirjaa. Nämä kirjat keskittyvät käytettävyyteen ja niissä on joitakin lukuja, jotka keskittyvät siihen, mitä käyttöliittymän suunnittelussa on otettava huomioon.

Ensimmäinen kirjalähde on ”Käytettävyys, suunnittelu ja arviointi” -kirjan luku viisi: ”Visuaalisen suunnittelun perusteita” (Kuutti 2003, 90-103). Toinen kirjalähde on ”Käytettävyyden psykologia” -kirjan luvut kuusi ja seitsemän: ”Havaitseminen ja tuotteen käyttö”, sekä ”Vuorovaikutus tuotteen kanssa” (Kuoppala ym. 2006, 58-141).

6.2 Suunnittelu

Tässä kappaleessa kerrotaan käyttöliittymän ratkaisuista sekä perusteluita niihin. Tässä ei esitellä tallennus-toiminnon dialogia, koska toteutuksessa on käytetty Swingin omaa versiota tästä.

6.2.1 Päänäkymä

Sovelluksessa on pääikkuna, jossa käyttäjä tekee raportin. Tämän pääikkunan kautta käyttäjä käyttää sovellusta ja sen toimintoja.

Käyttäjän kannalta on paras, jos hän näkee vain ne asiat, joita hän sillä hetkellä tarvitsee (Liite 1, Nielsenin lista 1). Tästä syystä käyttöliittymän pääikkunassa näkyy vain taulukko. Ikkunan yläreunassa on myös valikko, jonka kautta käyttäjä pääsee käsiksi sovelluksen toimintoihin.

Pääikkunan taulukko ei kuitenkaan näy heti, vaan taulukko luodaan vasta sen jälkeen, kun tiedot on ladattu tietokannasta. Tämä siksi, koska sovelluksen tarkoituksena on luoda raportti, joka pohjautuu tietokannasta ladattuihin tietoihin. Tästä syystä käyttäjällä ei ole tarvetta taulukolle, ennen kuin tiedot on ladattu tietokannasta.

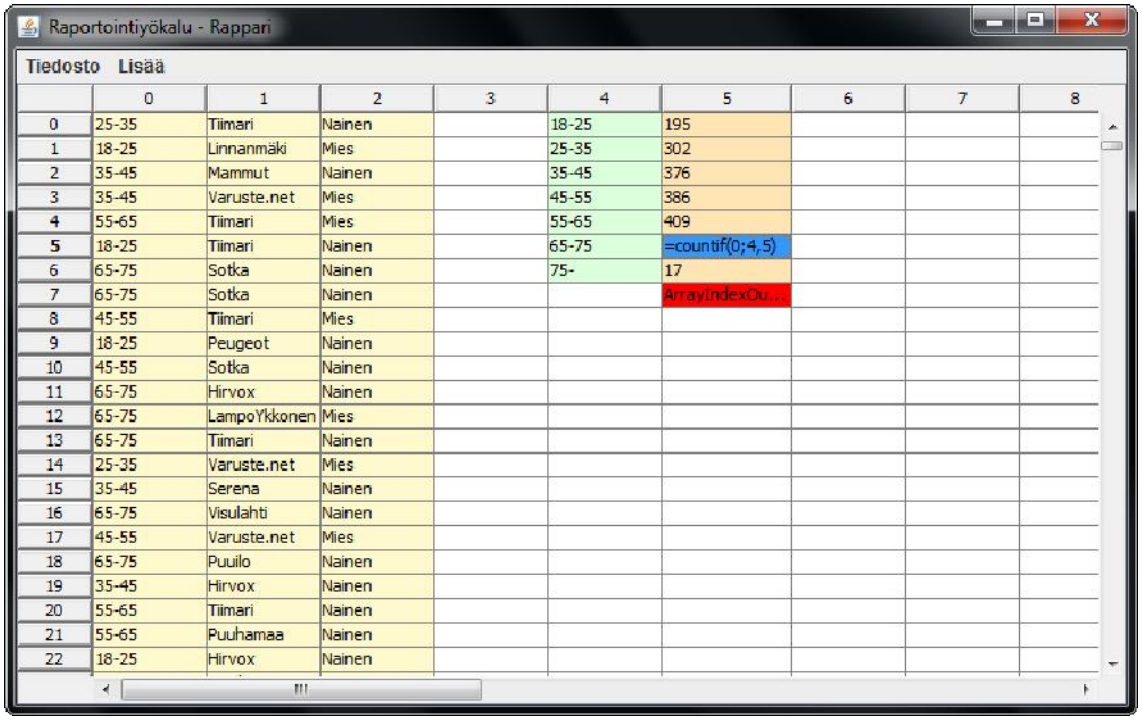
Taulukossa on erilaisia tietoja, kuten listoja, tietokannasta ladattuja tietoja ja kaavoja. Käyttäjän voi olla nopealla vilkaisulla hankala hahmottaa raportin rakennetta, varsinkin jos raportissa on paljon tietoja. Raportin rakenteen hahmottamisella tarkoitetaan sitä, että käyttäjä näkee, millaisia tietoja soluissa on kokonaisvaltaisesti. Käyttäjän olisi hyvä voida hahmottaa raportin rakenne nopealla vilkaisulla, jotta käyttäjällä ei kulu liikaa aikaa rakenteen hahmottamiseen vaan hän näkee suoralta kädeltä, mistä sarakkeesta kaavat puuttuvat.

Niin sanottujen hahmolakien avulla voidaan viestittää käyttäjälle, mitkä taulukkojen solut liittyvät toisiinsa. Hahmolakien mukaan lähellä toisiaan olevat, samankaltaiset, yhteen liitetyt tai rajatut objektit ovat yhteen kuuluvia. Helpoin tapa käyttää näitä hahmolakeja taulukon solujen kanssa on käyttää värejä. (Kuutti 2003, 28.)

Tässä sovelluksessa on käytetty taulukon solun taustaväriä hahmottamaan solun sisällön tyyppiä. Kuvassa 4 on kuvakaappaus valmiin sovelluksen päänäkymästä (kuva 4). Taulukossa on tietokannasta ladatut tiedot, lista ja kaavoja. Käyttäjä esimerkiksi näkee keltaisesta taustaväristä, että solu sisältää tietokannasta ladattuja tietoja, jolloin kyseisen solun tietoja ei voi muokata.

Taulukossa käytetyt värit eivät kuitenkaan saa olla räikeitä. Liian räikeät värit vievät toistuvasti käyttäjän huomion ja ärsyttävät häntä. Käyttäjän kannalta on paras, kun käyttöliittymän värit ovat yleisesti vain hillittyjä ja räikeitä jos käyttäjän huomio halutaan kiinnittää. (Kuutti 2003, 101.)

Taulukossa olevat kaavat ratkaistaan ja niiden tulos näytetään käyttäjälle, jotta hän voi paremmin hahmottaa, onko raportissa oikeat kaavat. Kaavojen ratkaisut näytetään samoin kuin Excelissä, eli kun kaava on kirjoitettu, solun teksti muuttuu kaavan ratkaisuksi. Valitsemalla solun käyttäjä näkee, mikä kaava solussa on, ja kaksoisklikkaamalla solua käyttäjä voi muokata kaavaa normaalisti.



	0	1	2	3	4	5	6	7	8
0	25-35	Tiimari	Nainen		18-25	195			
1	18-25	Linnanmäki	Mies		25-35	302			
2	35-45	Mammut	Nainen		35-45	376			
3	35-45	Varuste.net	Mies		45-55	386			
4	55-65	Tiimari	Mies		55-65	409			
5	18-25	Tiimari	Nainen		65-75	=countif(0;4,5)			
6	65-75	Sotka	Nainen		75-	17			
7	65-75	Sotka	Nainen			ArrayIndexOu...			
8	45-55	Tiimari	Mies						
9	18-25	Peugeot	Nainen						
10	45-55	Sotka	Nainen						
11	65-75	Hirvox	Nainen						
12	65-75	LampoYkkonen	Mies						
13	65-75	Tiimari	Nainen						
14	25-35	Varuste.net	Mies						
15	35-45	Serena	Nainen						
16	65-75	Visulahti	Nainen						
17	45-55	Varuste.net	Mies						
18	65-75	Puuilo	Nainen						
19	35-45	Hirvox	Nainen						
20	55-65	Tiimari	Nainen						
21	55-65	Puuhamaa	Nainen						
22	18-25	Hirvox	Nainen						

Kuva 4: Sovelluksen päänäkymä

Sovelluksen toiminnot ovat käytettävissä valikon kautta. Valikko on sijoitettu pääikkunan yläreunaan, kuten käytännössä kaikissa sovelluksissa. Tästä poikkeaminen aiheuttaa turhaa hämmennystä käyttäjälle, sillä käyttäjä lähtökohtaisesti olettaa valikon sijaitsevan ikkunan yläreunassa.

Joidenkin valikon vaihtoehtojen valintamahdollisuuksia on rajoitettu. Nämä rajoitukset ovat voimassa ennen tietojen lataamista tietokannasta, koska käyttäjällä ei ole näille toiminnoille tarvetta ennen sitä. Rajoitettu vaihtoehto näkyy valikossa harmaalla tekstillä (Ks. kuva 5).

Jotkut valikon vaihtoehdoista eivät liity juurikaan toisiinsa. Tästä syystä käytetään erotinta, jonka avulla viestitään käyttäjälle mitkä vaihtoehdot kuuluvat samaan kategoriaan. Kuvassa 5 on laajennettu Lisää-valikko. Esimerkiksi ”Sarake ja ”Rivi”-vaihtoehdot kuuluvat samaan kategoriaan, mutta ”Lista” ja ”Kaava” eivät kuulu tähän. Tästä syystä nämä ovat omissa osioissaan.

Kuva 5: Laajennettu Lisää-valikko ennen ja jälkeen tietojen latauksen

6.2.2 Tietojen valinta -dialogi

Jotta sovelluksen käyttö olisi mahdollisimman yksinkertaista, käyttäjälle olisi parasta näyttää vain ne toiminnot, jotka hän sillä hetkellä tarvitsee (Liite 1, Nielsenin lista 1). Tästä syystä tietojen valinta tapahtuu erillisessä dialogissa, sillä käyttäjällä ei ole tälle toiminnallisuudelle jatkuvaa tarvetta sovelluksessa. (Kuutti 2003, 50.)

Tämän dialogin kautta valitaan kolme tietojen latauksen kannalta oleellista asiaa; sarakkeet, aikaväli ja ehdot. Näistä sarakkeet ja aikaväli ovat pakollisia tietoja ja ehdot vapaaehtoisia. Tästä syystä ehtojen valinta tapahtuu omassa dialogissaan. Ehtojen valinta -dialogista kerrotaan tarkemmin myöhemmin tässä kappaleessa.

Kuvassa 6 on kuvakaappaus valmiin sovelluksen tietojen valinta –dialogista. Ylhäältä alas luettaessa, asiat etenevät niiden tärkeysjärjestyksen mukaan.

Sarakkeiden valinta tapahtuu rastittamalla sarakkeen nimen vieressä oleva valintaruutu. Sarakkeiden nimet ja valintaruudut on laitettu taulukkoon, sillä taulukko auttaa dialogin raken-

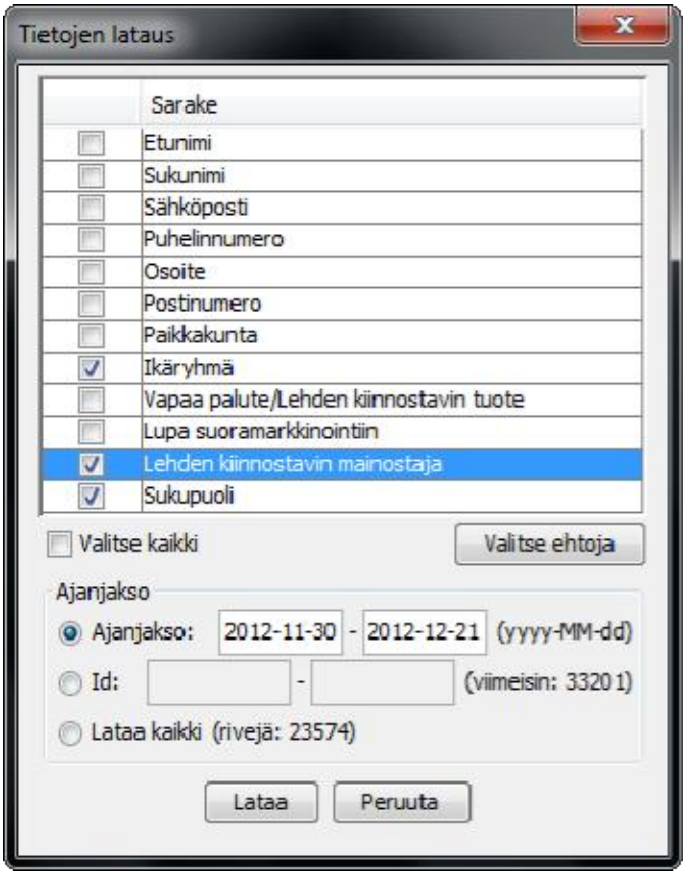
teen hahmottamisessa selkeästi rajaamalla sarakkeet omaksi kokonaisuudeksi. Taulukon alapuolella on valitse kaikki -valintaruutu, jonka avulla käyttäjä voi valita kaikki sarakkeet yhdellä klikkauksella.

Valitse kaikki -valintaruudun tasolla dialogin oikeassa reunassa on valitse ehtoja –painike. Painiketta painamalla avautuu valitse ehtoja -dialogi. Tämän painikkeen sijainti viestii käyttäjälle siitä, että tämä painike liittyy sarakkeisiin.

Aikavälin valinta tapahtuu valitsemalla yksi käytettävissä olevista valintanapeista. Nämä valintanapit on ympäröity otsikoidulla rajauksella. Rajauksen tarkoituksena on viestiä käyttäjälle, että nämä valintanapit liittyvät toisiinsa. Otsikko taas kertoo käyttäjälle, mihin valintanapin valinta vaikuttaa. (Kuutti 2003, 28.)

Aikavälin vaihtoehtoina ovat: lataa kaikki; aikaväli ja rivin id. Lataa kaikki –vaihtoehto on oletuksena valittu, sillä tätä vaihtoehtoa varten ei tarvitse täyttää erillisiä tietoja. Käyttäjän halutessa ladata kaikki rivit olisi käytännöllistä, jos käyttäjän ei tarvitsisi erikseen valita lataa kaikki –vaihtoehtoa. Jos käyttäjä haluaisi valita jonkin tietyn aikavälin, hänen täytyisi joka tapauksessa kirjoittaa haluamansa aikaväli.

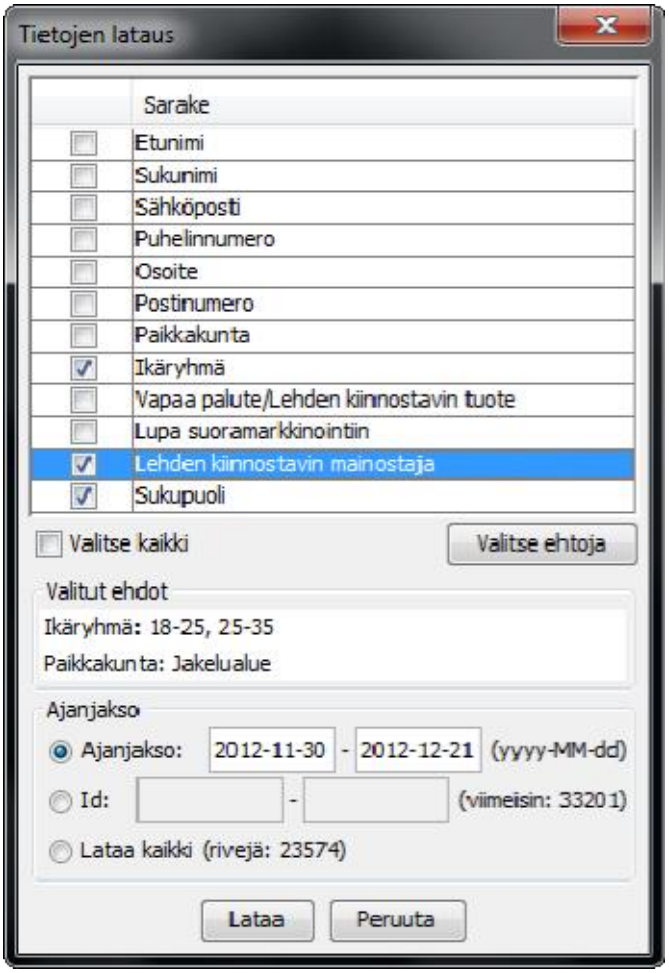
Käyttäjää pyritään opastamaan mahdollisimman paljon aikavälin valinnassa. Päivämäärävaihtoehdon perässä on teksti ”yyyy-MM-dd”. Tämän tarkoituksena on kertoa käyttäjälle, miten päivämäärät tulisi antaa sovellukselle, sillä sovellus hyväksyy päivämäärän vain tietyssä muodossa. Id-vaihtoehdon perässä on luku, joka kertoo, mikä on viimeisimmän rivin id. Lataa kaikki -vaihtoehdon perässä on luku, joka kertoo, kuinka monta riviä tietokannassa on. Tämän avulla käyttäjä pystyy hahmottamaan kuinka iso raportista tulisi.



Kuva 6: Tietojen valinta -dialogi

Käyttäjän on mahdollista valita ehtoja erillisessä dialogissa. Käyttäjän voi kuitenkin olla vaikea muistaa, mitä kaikkia ehtoja hän on valinnut, varsinkin jos hän muokkaa aikaväliä tai hänen työntekonsa keskeytyy. Tästä syystä, jos käyttäjä on valinnut ehtoja, valitut ehdot näkyvät tässä dialogissa. Kuvassa 7 on kuvakaappaus samaisesta dialogista, mutta tällä kertaa käyttäjä on valinnut ehtoja.

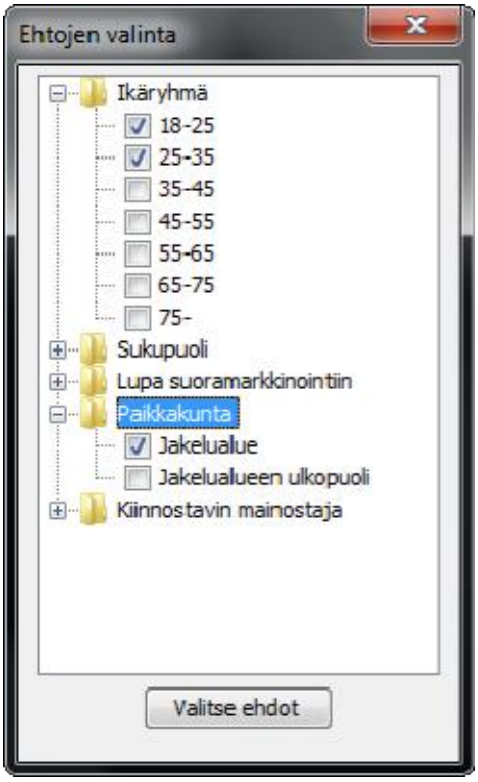
Nämä ehdot sijaitsevat Valitse ehtoja -painikkeen alapuolella, jotta käyttäjä yhdistäisi nämä toisiinsa. Ehdot on jaoteltu sen sarakkeen mukaan, johon ehto vaikuttaa. Näin käyttäjä pystyy helposti hahmottamaan valitsemiensa ehtojen vaikutuksia. Tämä ominaisuus tukee myös hyvin käyttäjän muistin kuormittamisen kieltoa (Liite 1, Nielsenin lista 3).



Kuva 7: Tietojen valinta -dialogi, kun ehtoja on valittu

Kuten aiemmin todettiin, ehtojen valinta tapahtuu omassa dialogissaan. Tässä pätee sama kuin kappaleen alussa todettiin: tätä toimintoa on turha näyttää, jos käyttäjä ei sitä tarvitse. Kuvassa 8 on kuvakaappaus valmiin sovelluksen valitse ehtoja -dialogista.

Ehdot on lajiteltu puurakenteeseen sen mukaan, mihin sarakkeeseen ehto vaikuttaa. Näin käyttäjän on helppo hahmottaa, mikä ehto vaikuttaa mihinkin, sillä käyttäjä näkee suoraan tästä hierarkkisesta asettelusta, mihin sarakkeeseen ehto kuuluu. Ehtojen valinta tapahtuu rastittamalla ehdon nimen edessä oleva valintaruutu. (Kuoppala ym. 2006, 139.)



Kuva 8: Ehtojen valinta -dialogi

6.2.3 Kaavan lisäys -dialogi

Kaavan lisäys -dialogi on tarkoitettu suurien kaavamäärien automaattiseen lisäykseen. Toiminto toimii niin, että käyttäjä antaa dialogin kautta sovellukselle tarvittavat tiedot ja sovellus lisää kaavat automaattisesti.

Tällä dialogilla on myös ominaisuus, jonka avulla käyttäjä voi esimerkiksi kirjoittaa taulukkoon ja dialogi pysyy silti päällimmäisenä. Tästä on hyötyä silloin, kun käyttäjä huomaa, että hän ei muista, mihin kaavat piti lisätä ja hänen täytyy vierittää taulukkoa alaspäin. Yleensä dialogeilla ei ole tällaista ominaisuutta, jolloin ne menevät sovelluksen alle ”piiloon”, jos pääikkunaa klikataan. Dialogeilla ei ole tehtäväpalkissa omaa tehtävää, jolloin dialogi pitäisi etsiä manuaalisesti pääikkunan alta.

Tätä dialogia suunniteltaessa on lähdetty siitä ajatuksesta, että käyttäjä on luomassa pystysuuntaista raporttia. Raporttien rakenteessa tyypillistä on se, että sarakkeessa yksi on allekkain tiedot siitä, millaista tietoa vastaavalla rivillä sarakkeessa kaksi on jotakin sarakkeeseen yksi liittyvää tietoa. Esimerkiksi solussa A1 on teksti ”Summa” ja solussa B1 on luku 58. Käyttäjä olettaisi näiden liittyvän toisiinsa.

Dialogin on oltava mahdollisimman yksiselitteinen, jotta käyttäjä ymmärtää, miten kaavan lisäys toimii. Kuvassa 9 on kuvakaappaus valmiin sovelluksen kaavan lisäys -dialogista.

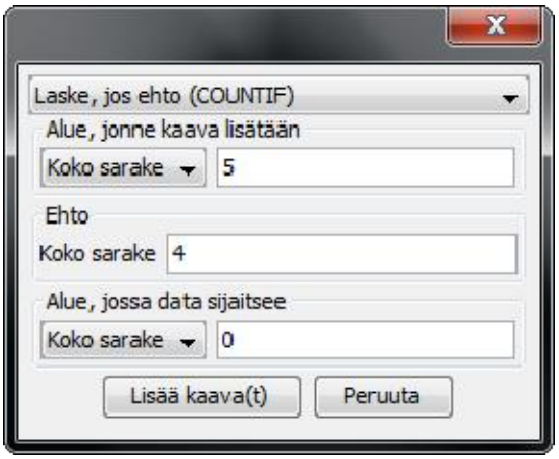
Dialogin alussa käyttäjä valitsee pudotusvalikosta kaavan, jonka hän haluaa lisätä taulukkoon. Kaavan valinta vaikuttaa niihin tietoihin, joita käyttäjän on annettava sovellukselle. Esimerkiksi SUM-kaavalle on annettava tiedot sijainti ja alue. COUNTIF-kaavan tapauksessa annetaan samat tiedot ja näiden lisäksi ehtojen sijainti. PERCENT-kaavalle on annettava sijainti, jaettava ja jakaja.

Sijainnilla tarkoitetaan, sitä mihin kaavat lisätään. Käyttäjä voi määritellä sijainniksi tietty solu, tietty alue tai koko sarake.

Alueella tarkoitetaan, missä kaavan tarvitsemat tiedot sijaitsevat. Käyttäjä voi määritellä alueeksi tietyn alueen tai koko sarakkeen.

Ehdon sijainnin ja jaettavan kohdalla käyttäjä voi antaa ainoastaan sarakkeen tiedon. Sovellus katsoo rivin tiedon samaksi kuin kyseessä olevan kaavan rivi, koska oletettavasti käyttäjä haluaa, että nämä tiedot yhdistetään toisiinsa valmiissa raportissa.

Jakajan kohdalla käyttäjä voi antaa ainoastaan tietyn solun tiedon, koska jakajan sijainti on vakio. Lähtökohtaisesti on ajateltu, että yhdellä lisäyskerralla lisätään tiettyyn jakajaan kohdistuvat kaavat.



Kuva 9: Lisää kaava -dialogi

6.2.4 Navigointi

Nielsenin listassa kohdassa seitsemän todetaan, että oikopolkuja ja tehokasta työskentelyä tulisi tukea (Liite 1, Nielsenin lista 7). Näillä oikopoluilla tarkoitetaan esimerkiksi erilaisia

näppäinyhdistelmiä tai hiiren kaksoisklikkausta. Oikopolkujen ansiosta sovelluksen käytettävyyttä ja tehokasta työskentelyä tuetaan, kun käyttäjän aikaa ei tuhlaannu oikean toiminnon etsimiseen valikosta vaan hän voi vain käyttää oikopolkua, jolloin toiminto aktivoituu.

Tässä sovelluksessa on käytetty pääsääntöisesti näppäinyhdistelmiä oikopolkuina. Näppäinyhdistelmät ovat muotoa Ctrl ja jokin kirjain. Esimerkiksi uuden tietokantahaun näppäinyhdistelmä on Ctrl+N. Näppäinyhdistelminä on pyritty käyttämään samoja näppäinyhdistelmiä kuin Windowsissa toimivien sovellusten vastaavien toimintojen näppäinyhdistelmät.

Käyttäjän on myös mahdollista navigoida valikossa näppäimistön avulla. Näppäimistöllä navigointi tarkoittaa sitä, että käyttäjä navigoi sovelluksessa hiireen koskematta. Kaikkiin valikon vaihtoehtoihin ei voida liittää oikopolkua, joten näppäimistöllä navigointi on seuraava vaihtoehto. Varsinkin kokeneet käyttäjät navigoivat näppäimistön kautta, jos halutulla toiminnolla ei ole oikotietä. Aluksi valikko fokusoidaan Alt-näppäintä painamalla. Tämän jälkeen käyttäjä voi navigoida valikossa nuoli- tai kirjainnäppäimillä.

6.2.5 Virheilmoitukset

Sovellusta käytettäessä sattuu silloin tällöin virhetilanteita ja näissä tilanteissa käyttäjän on saatava tieto tapahtuneesta virheestä. Lähtökohtaisesti virhetilanteiden muodostuminen pyritään estämään, mutta aina se ei ole mahdollista (Liite 1, Nielsenin lista 9). Virhetilanne ei myöskään saa johtaa sovelluksen kaatumiseen. (Kuutti 2003, 62-64.)

Virheilmoituksen sisällön on oltava selkeä, jotta käyttäjä ymmärtää, miksi virhetilanteeseen on päädytty (Liite 1, Nielsenin lista 8). Käyttäjälle on ilmoitettava, mistä virhe johtuu ja mahdollisesti myös se, miten tilanne korjataan tai estetään. Tällaisella rakentavalla ilmoituksella käyttäjä voi tulevaisuudessa välttää virhetilanteen kokonaan. Käyttäjää ei myöskään saa syyllistää viestissä, jotta käyttäjän käyttökokemus säilyy positiivisena. (Kuutti 2003, 61-62.)

Dialogeissa tapahtuvat virhetilanteet näytetään ponnahdusikkunoissa. Ponnahdusikkunoita käytetään nimenomaan dialogeissa, koska käyttäjän saattaisi olla vaikea huomata dialogiin ilmestyvää virheilmoitusta.

Ponnahdusikkunan avulla on helppo kiinnittää käyttäjän huomio. Sen avulla voidaan myös estää käyttäjää etenemästä.

Sovelluksessa on eräänlainen piilotettu virheilmoitus. Käyttäjä voi syöttää virheellisen kaavan tai käytettäessä SUM-kaavoja on mahdollista, että syntyy ikuinen silmukka, jossa kaksi SUM-kaavaa laskee toisiaan yhteen toistuvasti. Näissä tilanteissa kaavoja ei luonnollisesti lasketa,

koska se johtaisi sovelluksen kaatumiseen, vaan käyttäjälle ilmoitetaan tilanteesta samaan tapaan kuin Excelissä.

Excel ilmoittaa käyttäjälle virheellisestä kaavasta näyttämällä solun tekstinä VIRHE sekä virheen koodin. Tässä sovelluksessa solun teksti muutetaan myös muotoon Virhe, mutta tämä yksin ei takaa käyttäjän huomiota. Tästä syystä virhetilanteen aiheuttavan kaavan solun taustaväri muutetaan punaiseksi. Punainen väri poikkeaa vahvasti muusta ympäristöstä ja käyttäjä yhdistää punaisen värin ongelmatilanteeseen. Tällöin käyttäjä hyvin todennäköisesti kiinnittää huomionsa tähän soluun. (Kuutti 2003, 93.)

7 Toteutus

Tässä luvussa kerrotaan, miten tietyt sovelluksen ominaisuudet on toteutettu. Lukuun on pyritty ottamaan niitä asioita, jotka tuovat lisäarvoa käyttäjälle.

7.1 Käyttöliittymän ulkoasu

Kuten arkkitehtuurikuvauksessa kerrottiin, sovellus toteutetaan Javan Swingillä. Swing-komponentit toteutetaan kokonaan Javalla, joten ne näyttävät samalta käyttöliittymästä riippumatta. Tämä saattaa kuitenkin olla pelottavaa käyttäjälle, joka on tottunut siihen, että sovelluksen komponentit näyttävät tietynlaiselta.

Swingillä on Look&Feel -rajapinta, jonka avulla komponenttien ulkoasua voidaan muuttaa. Tällä rajapinnalla voidaan muuttaa komponenttien ulkoasua jonkin tietyn käyttöjärjestelmän mukaiseksi, esimerkiksi Windowsin mukaiseksi. On myös mahdollista muuttaa komponentit käytössä olevan käyttöjärjestelmän mukaiseksi, jolloin komponenttien ulkoasua riippuisi käyttöjärjestelmästä. Tässä sovelluksessa on tehty jälkimmäinen, jotta käyttäjä ei hämääntyisi oudon näköisestä käyttöliittymästä.

7.2 Taulukko

Edellisessä luvussa kerrottiin, kuinka eri tilanteet vaikuttavat solujen väriin ja tekstiin. Tässä kappaleessa esitellään, kuinka tämä ominaisuus on saatu toteutettua.

Sovelluksen taulukko on toteutettu Swingin JTable-komponentilla. JTablen ulkoasuun voidaan vaikuttaa renderoijien avulla. Renderoija on komponentti, joka muuttaa solun ulkonäköä niin, että solun sisältö pysyy samana, mutta käyttäjälle näytetään eri sisältö. Tämä toimii niin, että renderoijalle annetaan ehtoja, joiden mukaan solun ulkoasua vaihdetaan. Esimerkiksi voi-

daan antaa ehto, että jos solun teksti on ”Helsinki” solun taustaväri vaihdetaan oranssiksi. (Oracle 2010b.)

7.2.1 Solun ulkoasun muuttaminen

Edellisessä luvussa kerrottiin kuinka solun väri riippuu siitä sisältääkö se tietokannasta ladattuja tietoja, listoja tai kaavoja. Jotta sovellus tunnistaisi, millaista sisältöä on missäkin solussa, sovelluksessa on cellType-niminen taulukko. Tämä taulukko sisältää tiedon siitä, onko solun sisältö kaava, lista tms.

Renderoija päivittää taulukon ulkoasun aina kun taulukon sisältöön tehdään muutoksia. Tällöin renderoija käy jokaisen solun erikseen läpi ja muuttaa solun värin vastaamaan sisällön tyyppiä.

Taulukon kaavat lasketaan ja tulos näytetään käyttäjälle, jotta käyttäjä hahmottaisi, ovatko kaikki tarvittavat kaavat lisätty raporttiin. Renderoijan päivittäessä taulukkoa, se pyytää erillistä skriptiä laskemaan kaavat ja näyttää käyttäjälle saadun tuloksen. Sovellus kykenee ratkaisemaan niin funktioita kuin puhtaasti numeerisia laskutoimituksia.

Funktioilla tarkoitetaan SUM-, COUNTIF-, yms. kaavoja. Sovelluksella on skripti, joka tunnistaa nämä funktiot. Jos solu sisältää funktion, kyseinen kaava lähetetään eteenpäin toiselle skriptille. Tämä skripti laskee funktion ja kun se on laskettu, sen tulos annetaan renderoijalle.

Sovellus kykenee laskemaan myös puhtaasti numeerisia laskutoimituksia. Solussa oleva kaava on tekstimuodossa, joten Java ei itse kykene ratkaisemaan näitä kaavoja. Tähän on otettu apuun JavaScript. JavaScript alustetaan sovellukseen, jonka jälkeen sovellus pystyy käyttämään JavaScriptin funktioita. Numeeriset kaavat lasketaan JavaScriptin eval()-funktion avulla. (Refsnes Data 2012.)

Sovellus pystyy laskemaan myös monisisältöisiä kaavoja. Tällaisia kaavoja ovat ne, jotka sisältävät useita funktioita tai kaavoja jotka sisältävät sekä numeerisia laskutoimituksia että funktioita. Tämä toimii niin, että tarkistetaan, sisältääkö kaava funktioita. Jos näin on, funktio ratkaistaan ja se korvataan tuloksella. Tämän jälkeen kaava tarkistetaan funktioiden varalta uudelleen niin pitkään, kuin funktioita löytyy. Kun funktioita ei enää löydy, loput kaavasta ratkaistaan JavaScriptin avulla.

7.2.2 Virhetilanteet

Edellisessä luvussa kerrottiin, kuinka käyttäjälle näytetään kaavojen kohdalla tapahtuvat virhetilanteet. Renderoija tarkistaa jokaisen kaavan kohdalla, onko kaava virheellinen ja tarpeen vaatiessa käyttäjälle näytetään virheilmoitus. Kirjoitusvirheet, jotka estävät kaavan ratkaisun kokonaan, eivät ole kovin vaarallisia, joten niiden kohdalla riittää kaavan ratkaisun epäonnistumisesta kertova virheilmoitus.

Toiset virheet eivät estä kaavan ratkaisun yrittämistä, mutta sovelluksen yrittäessä ratkaista näitä kaavoja sovellus kaatuu. Näitä virheitä ovat esimerkiksi kaavan joutuminen ikuiseen silmukkaan tai viittaus soluun, jota ei ole olemassa. Näihin virheisiin pyritään varautumaan etukäteen. Näitä virheitä varten sovelluksessa on skriptejä, jotka tarkistavat johtaisivatko kaavojen ratkaisu näihin virhetilanteisiin. Mikäli kaavan ratkaisemisen havaitaan johtavan tällaiseen tilanteeseen, tieto tallennetaan errorTable-taulukkoon. Tähän taulukkoon tallennetaan virheen tyyppi ja renderoija tarkistaa tästä taulukosta onko kaava turvallista laskea.

7.3 Raportin tallennus

Edellisessä luvussa ei esitelty raportin tallennus -dialogin suunnittelua, koska sen toteutukseen on käytetty Swingin omaa JFileChooser-dialogikomponenttia. JFileChooserilla on mahdollista toteuttaa tiedoston avaava tai sulkeva dialogi. Tiedoston sulkevalla dialogilla tarkoitetaan dialogia, jonka avulla valitaan tiedoston tallennuspaikka ja nimi. Tässä sovelluksessa on käytetty vain sulkevaa dialogia, koska sovelluksella ei ole tarkoitettu olemassa olevien tiedostojen muokkaukseen. (Kosonen ym. 2007, 355.)

JFileChooser-dialogi on oletuksena englanninkielinen. Dialogin kieli on mahdollista lokalisoida eli elementtien tekstit voidaan muuttaa oman mieltymyksen mukaiseksi. Suomen kielelle ei ole kielipakettia, jonka avulla muutettaisiin koko dialogin kieli, joten dialogin elementtien tekstit on muutettava yksi kerrallaan.

Itse tiedoston kirjoittaminen tapahtuu niin, että sovellus käy taulukon solu kerrallaan läpi ja tallentaa solun sisällön tiedostoon. Sovelluksen on kuitenkin ensin selvitettävä, minkä tyyppistä sisältöä solu sisältää ja tallentaa ne oikeassa muodossa, koska kaikkia tietoja ei voi tallentaa tekstinä. Jos numeerinen tieto tai kaava tallennettaisiin tekstinä, Excel käsittelisi näiden solujen sisältöä tekstinä, eikä esimerkiksi ymmärtäisi ratkaista kaavoja.

Sovelluksen käyttämät kaavat on muutettava vastaamaan Excelin formaattia. Numeeriset kaavat vaativat vain pientä muuntelua, mutta sisäänohjelmoidut kaavat joudutaan muuttamaan lähes kokonaan. Esimerkiksi kaavojen sisältämien koordinaattien muuttaminen on vält-

tämätöntä, sillä Excel käyttää sarakkeiden tunnistamiseen kirjaimia, numeroiden sijaan. Sovelluksella on tätä varten skripti, joka muuttaa koordinaatit oikeaan muotoon.

Esimerkiksi sovelluksen sisäänohjelmoitu prosenttikaava on kirjoitettu muotoon percent(2,0;3,0). Tämä muutetaan tiedostoa varten muotoon C1/D1 ja tallennetaan kaavana. Lisäksi sovellus tallentaa myös tiedon siitä, että kaavan tulosta tulee käsitellä prosenttilukuna.

Toisessa esimerkissä kaava on kirjoitettu taulukkoon muotoon sum(0). Tämä muutetaan muotoon SUM(A1:A#), missä # on taulukon rivien lukumäärä. Sovelluksen sisältämä skripti on tässä tapauksessa muuttanut koko sarakkeen kattavan kaavan suurimpaan mahdolliseen rajaan asti.

7.4 Kielitiedostot

Edellisessä kappaleessa (7.3) mainittu dialogin lokalisointi käyttää hyväkseen kielitiedostoa. Kielitiedostot ovat tiedostoja, jotka sisältävät käyttäjälle näytettävät tekstit. Kielitiedoston ansiosta sekä tekstien hallinnointi ja sovelluksen kielen muuttaminen on helppoa, sillä kaikki tekstit löytyvät samasta paikasta.

Kielitiedostot ovat properties-tiedostoja. Tiedoston rakenne toimii niin, että yhdellä rivillä on yksi avain ja tämän avaimen jälkeisen =-merkin jälkeen on teksti. Esimerkiksi AVAIN = Arvo. Avaimella tarkoitetaan uniikkia tekstiä, jonka pitäisi olla mahdollisimman kuvaava siitä, mihin teksti kuuluu.

Näitä arvoja haetaan erillisellä metodilla, joka osaa lukea näitä properties-tiedostoja. Metodille annetaan avainarvo, ja metodi hakee avainta vastaavan tekstin tiedostosta.

8 Testaus

Sovellusta on testattu toteutuksen aikana ja sovelluksen valmistumisen jälkeen. Toteutuksen aikaisen testauksen tavoitteena on ollut tarkistaa eri sovellusten osien toimivuus. Toteutuksen jälkeisen testauksen tavoitteena on ollut testata, toimiiko koko sovellus yhtenäisesti.

Toteutuksen aikaisessa testauksessa on jokaisen osan valmistumisen jälkeen testattu itse osan toimivuus sekä se, kuinka hyvin osa toimii sovellukseen yhteen liitettynä. Testauksessa osia on käytetty sekä oikeaoppisesti että virheellisesti. Näiden tarkoituksena on ollut varmistaa, että sovellus toimii oikein sekä normaali että virhetilanteissa.

Kaikkien osien valmistuttua testattiin käyttötarinoiden toimivuus. Näiden testien tarkoituksena on ollut tarkistaa, että sovellus toimii halutusti käyttötarinoiden tilanteissa.

Käyttötarinoiden testauksen lisäksi koko sovelluksen käyttöä on testattu mahdollisimman paljon, jotta kaikki löydettävissä olevat virheet tulisi paikannettua. Tämä on sisältänyt normaalien, harvinaisten ja virheellisten tilanteiden luomista.

9 Yhteenveto

Sovelluksen kehitystyö onnistui hyvin ja vastasi kaikkiin vaatimusmäärittelyn asettamiin vaatimuksiin. Yksi tärkeimmistä puutteista on kuitenkin se, että listojen muokkaukselle ei ole omaa dialogia, vaan ne täytyy muokata tiedostossa käsin. Tämä toiminto on tarkoitus toteuttaa sovelluksen seuraavaan versioon.

Sovelluksen toteutus on ollut hyvin mielenkiintoista ja tarjonnut haasteita, joiden avulla olen voinut kehittää omaa osaamistani ohjelmoijana. Kehitystyö on antanut myös mahdollisuuden tutustua tarkemmin sovelluskehitykseen.

Lähteet

Kirjalähteet

Haikkala I. & Mikkonen T. 2011. Ohjelmistotuotannon käytännöt. 12. uudistettu painos. Hämeenlinna:Talentum

Kosonen P., Peltomäki J. & Silander S. 2007. Java 2 Ohjelmoinnin peruskirja. 2. painos. Vantaa: Dark

Kuoppala H., Parkkinen J., Sinkkonen I. & Vastamäki R. 2006. Käytettävyyden psykologia. 3. uudistettu painos. Helsinki:Edita

Kuutti, W. 2003. Käytettävyys, suunnittelu ja arviointi. Helsinki: Talentum

Internet-lähteet

JExcelApi. 2012. JExcelApi. Viitattu 14.9.2012. http://jexcelapi.sourceforge.net/

Oracle. 2012. MySQL Connector/J. Viitattu 20.12.2012.
http://dev.mysql.com/doc/refman/5.1/en/connector-j.html

Oracle. 2010. Java AWT Api. Viitattu 29.1.2013.
http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/package-summary.html

Oracle. 2010. Java Swing Api. Viitattu 29.1.2013.
http://docs.oracle.com/javase/1.4.2/docs/api/javax/swing/package-summary.html

Refsnes Data. 2012. JavaScript Tutorial. Viitattu 28.12.2012.
http://www.w3schools.com/js/default.asp

Kuvat

Kuva 1: Toimintakaavio sovelluksen toiminnoista 13
Kuva 2: Esimerkki useamman kaavan lisäyksen lopputuloksesta 15
Kuva 3: Sovelluksen arkkitehtuuri 17
Kuva 4: Sovelluksen päänäkymä 20
Kuva 5: Laajennettu Lisää-valikko ennen ja jälkeen tietojen latauksen 21
Kuva 6: Tietojen valinta -dialogi 23
Kuva 7: Tietojen valinta -dialogi, kun ehtoja on valittu 24
Kuva 8: Ehtojen valinta -dialogi 25
Kuva 9: Lisää kaava -dialogi 26

Liitteet

Nielsenin lista..... 36

Nielsenin lista

1. Vuorovaikutuksen käyttäjän kanssa tulee olla yksinkertaista ja luonnollista.
2. Vuorovaikutuksessa tulee käyttää käyttäjän kieltä.
3. Käyttäjän muistin kuormitus tulee minimoida.
4. Käyttöliittymän tulee olla yhdenmukainen
5. Järjestelmän tulee antaa käyttäjälle kunnollista palautetta reaaliajassa.
6. Ohjelmassa ja sen osissa tulee olla selkeät poistumistiet
7. Oikopolkuja ja tehokasta työskentelyä tulisi tukea.
8. Virheilmoitusten tulee olla selkeitä ja ymmärrettäviä.
9. Virhetilanteisiin joutumista tulisi välttää.
10. Käyttöliittymässä tulee olla kunnolliset avustustoiminnot ja dokumentaatio.